AMENDMENTS TO THE CLAIMS

1. (Currently amended) A method of preventing an attack on a network, wherein the attack comprises injecting a spurious transmission control protocol (TCP) segment into a TCP connection between a sender and a receiver, the method comprising the computer-implemented steps of:

receiving a duplicate TCP ACK message;

incrementing a false duplicate ACK counter when a TCP re-transmission buffer maintained by the receiver is empty;

when the false duplicate ACK counter is equal to a specified strike factor, and only when a TCP re-transmission buffer maintained by the receiver is empty, sending a corrective ACK message that provides a correct sequence value and ACK value.

2. (Original) A method as recited in Claim 1, wherein the specified strike factor is a value in a range of 1 to 10.

3. (Original) A method as recited in Claim 1, wherein the steps are performed by an endpoint node acting as the receiver of data in the TCP connection.

4. (Original) A method as recited in Claim 1, wherein the steps are performed by a TCP application of an operating system of a network infrastructure element.

5. (Original) A method as recited in Claim 1, wherein the steps are performed by a TCP process, stack, adapter or agent hosted by or associated with an operating system of a personal computer, workstation or other network end station.

6. (Original) A method as recited in Claim 1, further comprising the steps of:

receiving the corrective ACK message;

determining whether the correct sequence value is less than another sequence value of a segment in a re-assembly buffer;

discarding the segment from the re-assembly buffer when the correct sequence value is less than the other sequence value of a segment in the re-assembly buffer.

7. (Original) A method as recited in Claim 6, wherein the discarding step comprises discarding all segments in the re-assembly buffer.

8. (Original) A method of preventing an attack on a network, wherein the attack comprises injecting a spurious transmission control protocol (TCP) segment into a TCP connection between a sender and a receiver, the method comprising the computer-implemented steps of:

receiving a particular TCP segment;

determining a sequence value gap as between the particular TCP segment and a prior TCP segment previously placed in a re-assembly buffer maintained by the receiver;

determining whether the sequence value gap is too large according to a specified heuristic;

if the sequence value gap is too large, then performing the steps of:

creating and sending a dummy segment carrying a particular sequence value that is just prior to a last properly acknowledged sequence value;

receiving an acknowledgment of the dummy segment;

determining whether a second sequence value carried in the acknowledgment is less than a third sequence value of the first TCP segment; and

discarding the particular TCP segment from the re-assembly buffer when the second sequence value carried in the acknowledgment is less than the third sequence value of the particular TCP segment.

9. (Original) A method as recited in Claim 8, wherein the specified heuristic holds when a re-assembly gap value is greater than one-half of a then-current window size.

10. (Original) A method as recited in Claim 8, wherein the specified heuristic holds when a re-assembly gap value is greater than the lesser of (a) one-half of a then-current window size or (b) a multiple of a maximum allowed segment size.

11. (Original) A method as recited in Claim 8, wherein the specified heuristic holds when a query threshold is greater than a function of then-current bandwidth and then-current delay.

12. (Original) A method as recited in Claim 8, wherein the specified heuristic holds when a query threshold is greater than a function of a round-trip time (RTT) estimate, a then-current available bandwidth value, and window size.

13. (Original) A method as recited in Claim 8, wherein the specified heuristic holds when a query threshold value is less than or equal to a re-assembly gap value and the re-assembly gap value is less than or equal to a receive window value.

14. (Original) A method as recited in Claim 13, wherein the re-assembly gap value comprises a sequence value of a first segment in the re-assembly buffer less a next expected segment value.

15.-17. (Canceled)

18. (New) An apparatus for preventing an attack on a network, wherein the attack comprises sending a spurious transmission control protocol (TCP) segment with a spurious or unwanted DATA segment, the apparatus comprising:

means for receiving a duplicate TCP ACK message;

means for incrementing a false duplicate ACK counter when a TCP re-transmission buffer maintained by the receiver is empty;

means sending a corrective ACK message that provides a correct sequence value and ACK value when the false duplicate ACK counter is equal to a specified strike factor and only when a TCP re-transmission buffer maintained by the receiver is empty.

19. (New) An apparatus for preventing an attack on a network, wherein the attack comprises sending a spurious transmission control protocol (TCP) segment with spurious or unwanted DATA segment, comprising:

a processor;

one or more stored sequences of instructions that are accessible to the processor and which, when executed by the processor, cause the processor to perform:

receiving a duplicate TCP ACK message;

incrementing a false duplicate ACK counter when a TCP re-transmission buffer maintained by the receiver is empty;

when the false duplicate ACK counter is equal to a specified strike factor, and only when a TCP re-transmission buffer maintained by the receiver is empty, sending a corrective ACK message that provides a correct sequence value and ACK value.

20. (New) A computer-readable tangible storage medium carrying one or more sequences of instructions for preventing an attack on a network, wherein the attack comprises sending a spurious transmission-control protocol (TCP) segment with unwanted or spurious DATA, wherein the execution of the one or more sequences of instructions by one or more processors causes the one or more processors to perform:

receiving a duplicate TCP ACK message;

incrementing a false duplicate ACK counter when a TCP re-transmission buffer maintained by the receiver is empty;

when the false duplicate ACK counter is equal to a specified strike factor, and only when a TCP re-transmission buffer maintained by the receiver is empty, sending a corrective ACK message that provides a correct sequence value and ACK value.

21. (New) The apparatus of claim 18 or 19, wherein the specified strike factor is a value in a range of 1 to 10.

22. (New) The apparatus of claim 18 or 19, comprising an endpoint node acting as the receiver of data in the TCP connection.

23. (New) The apparatus of claim 18 or 19, comprising a TCP application of an operating system of a network infrastructure element.

24. (New) The apparatus of claim 18 or 19, comprising a TCP process, stack, adapter or agent hosted by or associated with an operating system of a personal computer, workstation or other network end station.

25. (New) The apparatus of claim 15, further comprising:
means for receiving the corrective ACK message;
means for determining whether the correct sequence value is less than another sequence value of a segment in a re-assembly buffer;
means for discarding the segment from the re-assembly buffer when the correct sequence value is less than the other sequence value of a segment in the re-assembly buffer.

26. (New) The apparatus of claim 16, further comprising sequences of instructions which when executed cause the one or more processors to perform:
receiving the corrective ACK message;
determining whether the correct sequence value is less than another sequence value of a segment in a re-assembly buffer;
discarding the segment from the re-assembly buffer when the correct sequence value is less than the other sequence value of a segment in the re-assembly buffer.

27. (New) The apparatus of claim 18 or 19, operable to discard all segments in the re-assembly buffer.

28. (New) An apparatus for preventing an attack on a network, wherein the attack comprises injecting a spurious transmission control protocol (TCP) segment into a TCP connection between a sender and a receiver, comprising:
means for receiving a particular TCP segment;

means for determining a sequence value gap as between the particular TCP segment and a prior TCP segment previously placed in a re-assembly buffer maintained by the receiver;

means for determining whether the sequence value gap is too large according to a specified heuristic;

means operable when the sequence value gap is too large for creating and sending a dummy segment carrying a particular sequence value that is just prior to a last properly acknowledged sequence value, for receiving an acknowledgment of the dummy segment, for determining whether a second sequence value carried in the acknowledgment is less than a third sequence value of the first TCP segment, and for discarding the particular TCP segment from the re-assembly buffer when the second sequence value carried in the acknowledgment is less than the third sequence value of the particular TCP segment.

29. (New) An apparatus for preventing an attack on a network, wherein the attack comprises injecting a spurious transmission control protocol (TCP) segment into a TCP connection between a sender and a receiver, comprising:

a processor;

one or more stored sequences of instructions that are accessible to the processor and which, when executed by the processor, cause the processor to perform:

receiving a particular TCP segment;

determining a sequence value gap as between the particular TCP segment and a prior TCP segment previously placed in a re-assembly buffer maintained by the receiver;

determining whether the sequence value gap is too large according to a specified heuristic;

when the sequence value gap is too large, creating and sending a dummy segment carrying a particular sequence value that is just prior to a last properly acknowledged sequence value, receiving an acknowledgment of the dummy segment, determining whether a second sequence value carried in the acknowledgment is less than a third sequence value of the first TCP segment, and discarding the particular TCP segment from the re-assembly buffer when the

second sequence value carried in the acknowledgment is less than the third sequence value of the particular TCP segment.

30. (New) The apparatus of claim 28 or 29, wherein the specified heuristic holds when a re-assembly gap value is greater than one-half of a then-current window size.

31. (New) The apparatus of claim 28 or 29, wherein the specified heuristic holds when a re-assembly gap value is greater than the lesser of (a) one-half of a then-current window size or (b) a multiple of a maximum allowed segment size.

32. (New) The apparatus of claim 28 or 29, wherein the specified heuristic holds when a query threshold is greater than a function of then-current bandwidth and then-current delay.

33. (New) The apparatus of claim 28 or 29, wherein the specified heuristic holds when a query threshold is greater than a function of a round-trip time (RTT) estimate, a then-current available bandwidth value, and window size.

34. (New) The apparatus of claim 28 or 29, wherein the specified heuristic holds when a query threshold value is less than or equal to a re-assembly gap value and the re-assembly gap value is less than or equal to a receive window value.

35. (New) The apparatus of claim 34, wherein the re-assembly gap value comprises a sequence value of a first segment in the re-assembly buffer less a next expected segment value.